1. **Tìm hiểu về tính đóng gói và tính trừu tượng.**

Tính đóng gói.

Tính đóng gói (Encapsulation) là một trong bốn tính chất cơ bản của lập trình hướng đối tượng trong Java.

Tính đóng gói là kỹ thuật ẩn giấu thông tin không liên quan và hiện thị ra thông liên quan. Mục đích chính của đóng gói trong java là giảm thiểu mức độ phức tạp phát triển phần mềm.

Tính đóng gói trong Java là một tiến trình đóng gói code và dữ liệu lại với nhau vào trong một đơn vị. Chúng ta có thể tạo một lớp được bao đóng hoàn toàn trong Java bằng việc tạo tất cả thành viên dữ liệu của lớp là private. Sau đó, chúng ta sử dụng phương thức setter và getter để thiết lập và lấy dữ liệu trong nó.

Tính đóng gói là kỹ thuật tạo một trường của lớp private và cung cấp khả năng truy cập trường này qua các phương thức pullic. Nếu một trường được khai báo là private, nó không thể được truy cập bởi bên ngoài lớp, do đó có thể che dấu các trường có lớp này. Vì lý do này, tính đóng gói được ám chỉ như việc dấu dữ liệu (data hiding).

Tính trừu tượng.

Tính trừu tượng là một tiến trình ẩn các chi tiết trình triển khai và chỉ hiển thị tính năng tới người dùng. Tính trừu tượng cho phép bạn loại bỏ tính chất phức tạp của đối tượng bằng cách chỉ đưa ra các thuộc tính và phương thức cần thiết của đối tượng trong lập trình.

Tính trừu tượng giúp bạn tập trung vào những cốt lõi cần thiết của đối tượng thay vì quan tâm đến cách nó thực hiện.

Trong Java, chúng là sử dụng **abstract class** và **abstract interface** để có tính trừu tượng.

2. **Các lợi ích của tính đóng gói và trừu tượng trong phát triển phần mềm**

- Bảo vệ dữ liệu: Tính đóng gói cho phép che giấu các chi tiết bên trong của một đối tượng, như các biến và phương thức. Chỉ có các phương thức public hoặc protected được cung cấp để tương tác với đối tượng. Điều này giúp bảo vệ dữ liệu của đối tượng và ngăn chặn truy cập trực tiếp và không kiểm soát đến các thành phần bên trong của đối tượng.
- Kiểm soát truy cập: Tính đóng gói cho phép kiểm soát cách mà các thành phần của đối tượng có thể được truy cập từ bên ngoài. Bằng cách xác định các phương thức public, private và protected, chúng ta có thể quản lý quyền truy cập vào dữ liệu và hành vi của đối tượng.
- Dễ bảo trì: Tính đóng gói giúp giảm thiểu sự phụ thuộc giữa các thành phần của một đối tượng. Nếu một thay đổi cần được thực hiện trong một thành phần cụ thể, chúng ta chỉ cần quan tâm đến các phương thức public và cách tương tác với đối tượng, trong khi các thành phần bên trong vẫn được bảo vệ.
- Tính tái sử dụng: Tính trừu tượng cho phép chúng ta tạo ra các lớp trừu tượng (abstract classes) và giao diện (interfaces), tạo ra một cơ sở chung để tái sử dụng mã và xây dựng các lớp con (subclasses). Điều này giúp giảm thiểu việc lặp lại mã và tạo ra một kiến trúc linh hoạt và mở rộng cho hệ thống phần mềm.
- Giảm sự phức tạp: Tính trừu tượng cho phép chúng ta tập trung vào các khía cạnh quan trọng và cung cấp một mức độ trừu tượng cao hơn cho người lập trình. Việc tạo ra các lớp trừu tượng và giao diện giúp giảm sự phức tạp của hệ thống và tạo ra một cấu trúc rõ ràng và dễ hiểu.

3. **Sự khác biệt, mối quan hệ của tính đóng gói và tính trừu tượng. Lợi ích của việc kết hợp hai tính chất trên trong việc phát triển phần mềm.**

Sự khác biệt, mối quan hệ.

- Tính trừu tượng chỉ hiển thị dữ liệu hữu ích bằng cách cung cấp các chi tiết cần thiết nhất. Trong khi tính năng Đóng gói bao bọc mã và dữ liệu để có thông tin cần thiết.
- Tính trừu tượng chủ yếu tập trung vào những gì nên làm. Trong khi Encapsulation tập trung vào cách nó nên được thực hiện.

- Tính trừu tượng che giấu sự phức tạp bằng cách cung cấp cho bạn một bức tranh trừu tượng hơn. Trong khi Encapsulation ẩn hoạt động bên trong để bạn có thể thay đổi nó sau này.
- Tính trừu tượng giúp bạn phân vùng chương trình thành nhiều phần độc lập. Trong khi tính năng Đóng gói dễ dàng thay đổi với các yêu cầu mới.
- Tính trừu tượng giải quyết vấn đề ở cấp độ thiết kế. Trong khi Tính đóng gói giải quyết vấn đề ở cấp độ thực thi.
- Tính trừu tượng che giấu các chi tiết không liên quan được tìm thấy trong mã. Trong khi tính năng Đóng gói giúp các nhà phát triển tổ chức toàn bộ mã một cách dễ dàng.

Lợi ích của việc kết hợp hai tính chất trên trong việc phát triển phần mềm.

- Tái sử dụng mã nguồn: Tính đóng gói và tính trừu tượng cung cấp một cách cấu trúc hóa codebase dễ tái sử dụng. Đối tượng được thiết kế với tính chất đóng gói, cho phép tái sử dụng code thông qua việc sử dụng lại các đối tượng đã được xây dựng trước đó. Đồng thời, tính trừu tượng cho phép tạo ra các lớp trừu tượng hoặc giao diện, cho phép sử dụng lại code một cách linh hoạt trong các ngữ cảnh khác nhau.
- Dễ bảo trì: Tính đóng gói giúp giấu thông tin nội bộ và hạn chế truy cập trực tiếp vào các thành phần của đối tượng. Điều này cho phép thay đổi cài đặt nội bộ mà không ảnh hưởng đến các phần khác trong hệ thống. Tính trừu tượng giúp tách biệt việc triển khai và sử dụng đối tượng, do đó, nếu có thay đổi trong triển khai, các đối tượng sử dụng không cần thay đổi. Điều này giúp giảm rủi ro gây lỗi và tạo điều kiện cho việc bảo trì và nâng cấp phần mềm một cách dễ dàng hơn.
- Tính linh hoạt và mở rộng: Kết hợp tính đóng gói và tính trừu tượng tạo ra một mô hình phát triển linh hoạt. Tính trừu tượng cho phép định nghĩa các khái niệm và giao diện trừu tượng, không phụ thuộc vào các chi tiết cài đặt cụ thể. Điều này giúp tăng tính linh hoạt và khả năng mở rộng của hệ thống, vì bạn có thể thêm hoặc thay đổi các đối tượng triển khai mà không ảnh hưởng đến sử dụng các đối tượng đó.

- Tăng tính rõ ràng và đơn giản: Việc sử dụng tính đóng gói và tính trừu tượng giúp rõ ràng hóa cấu trúc và mối quan hệ giữa các thành phần trong hệ thống. Đối tượng được thiết kế với các phương thức

công khai và các giao diện trừu tượng, đơn giản hóa việc sử dụng và tương tác với chúng. Điều này giúp cải thiện khả năng đọc, hiểu và bảo trì mã nguồn.

4. Cách thể hiện của 2 tính chất trong ngôn ngữ lập trình hướng đối tượng (java)

Tính đóng gói

- Sử dụng từ khóa private, public và protected để kiểm soát quyền truy cập vào các thành phần của một đối tượng, chẳng hạn như biến và phương thức.
- Sử dụng các getter và setter (phương thức truy cập) để đọc và ghi giá trị của các biến thành viên private, giúp bảo vệ dữ liệu và đảm bảo việc truy cập kiểm soát.
- Sử dụng các phương thức public để cho phép tương tác với các phần khác của hệ thống hoặc các đối tượng khác.

Tính trừu tượng

- Sử dụng lớp trừu tượng (abstract class) để tạo ra một lớp cơ sở có các phương thức trừu tượng mà các lớp con (subclasses) phải triển khai.
- Sử dụng giao diện (interface) để xác định các phương thức mà các lớp khác nhau phải triển khai.
- Sử dụng từ khóa abstract để khai báo phương thức trừu tượng trong lớp trừu tượng hoặc giao diện, mà không cần cung cấp cài đặt cụ thể cho phương thức đó.

Ví dụ:

```
public abstract class Shape {
  private String color;
  public Shape(String color) {
    this.color = color;
  }
  public String getColor() {
    return color;
  }
```

```
    public abstract double calculateArea();
}

public class Circle extends Shape {
    private double radius;
    public Circle(String color, double radius) {
        super(color);           //gọi constructor của lớp cha
        this.radius = radius;
    }
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}

public interface Drawable {
    void draw();
}

public class Rectangle implements Drawable {
    private int width;
    private int height;
    public Rectangle(int width, int height) {
        this.width = width;
        this.height = height;
    }
    public void draw() {
        System.out.println("Drawing a rectangle.");
    }
}
```

Lớp **Shape** là một lớp trừu tượng có phương thức trừu tượng **calculateArea**(). Lớp **Circle** là một lớp con của **Shape** và triển khai phương thức **calculateArea**() để tính diện tích của hình tròn.

Giao diện **Drawable** khai báo một phương thức trừu tượng **draw**(). Lớp **Rectangle** triển khai giao diện **Drawable** và cung cấp một cài đặt cho phương thức **draw**(), in ra thông báo vẽ một hình chữ nhật.